PATENT APPLICATION

GARBAGE COLLECTION WITH A DYNAMIC WINDOW

Inventors: Oleg Pliss & Bernd J. Mathiske

Related Application

This application claims priority to U.S. Provisional Application Serial No. 60/532,380, filed on December 23, 2003, which is hereby incorporated by reference.

BACKGROUND

This invention relates to the field of computer systems. More particularly, a system and method are provided for performing garbage collection with a dynamic young generation.

Many generational garbage collection schemes employing multiple generations suffer from over-promotion. That is, they promote too many memory objects from a younger generation to an older generation, or promote objects too frequently. As a result, the older generation must be garbage collected more frequently than it would otherwise.

One reason for over-promotion is that traditional generational garbage collection schemes work in closed environments, without considering or applying externally available information about the objects it handles. For example, garbage collectors generally do not have access to, or do not use, information concerning how likely or how soon a set of data objects will become garbage.

Some generational garbage collection schemes employ a fixed-size window that continually moves across a section of memory. The window defines

a younger generation of memory and data objects, while the area behind the moving window defines an older generation and contains data objects promoted from the younger generation because they survived a garbage collection performed on that generation. These schemes also suffer from over-promotion, because the window continually moves, and every object in the younger generation that survives a garbage collection is automatically promoted to the older generation, regardless of how likely or how soon the survivor will become garbage.

SUMMARY

In one embodiment of the invention, a system and methods are provided for performing intelligent generational garbage collection using a dynamic sliding window. During normal memory allocation, a sliding window defines a young generation within an older generation or other area of memory. When data are stored that will become garbage within a finite period of time, a temporary phase of operation is initiated.

In the temporary allocation phase, the lower bound of the window is fixed, while the upper bound is allowed to expand to accommodate new objects. When the data become garbage, or a predetermined period of time passes, the window is garbage collected and compacted, and normal memory allocation and garbage collection operations resume. Thus, the window is dynamic in both movement and size.

When the temporary allocation phase is initiated, the young generation may be garbage collected and compacted, and the lower window bound may be fixed at the location (e.g., address) where the allocation point was when the target data were stored.

In different embodiments of the invention, different events or operations may trigger or end the temporary allocation phase of operation. For example, when a compiler or Java Virtual Machine (JVM) begins to compile a series of instructions, during which a number of short-lived data objects will be used, a garbage collector may be signaled or instructed to start the temporary allocation phase. When compilation is complete, normal operations may be resumed. In other embodiments, the temporary allocation phase may be activated whenever a set of data will become garbage, or is likely to become garbage, within a relatively short period of time.

DESCRIPTION OF THE FIGURES

FIG. 1 is a block diagram of a generational garbage collection apparatus that uses a dynamic window for anticipatable garbage, according to one embodiment of the present invention.

FIG. 2 is a flowchart illustrating one method of performing garbage collection with a dynamic window for anticipatable garbage, according to one embodiment of the present invention.

DETAILED DESCRIPTION

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of particular applications of the invention and their requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art and the general principles defined herein may be applied to other embodiments and applications without departing from the scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is

to be accorded the widest scope consistent with the principles and features disclosed herein.

In an embodiment of the invention, a system and method are provided for intelligent generational garbage collection using a dynamically sized sliding window. Information about the nature of data objects in the memory tended by the garbage collector (e.g., a heap) is used to dynamically adjust the size and/or behavior of the sliding window.

In one implementation of this embodiment, when it is certain (or maybe even just likely) that a relatively large amount of data being used or about to be use will become garbage in a finite or predetermined period of time, the bottom of the window may stop moving, while the top of the window is allowed to expand to accommodate new objects. After the data becomes garbage and the young generation is garbage-collected and possibly compacted, the window may return to its normal size and behavior. Thus, as a result of temporarily sticking or holding the bottom of the window, the system avoids over-promotion and may reduce the frequency with which an older generation of memory needs to be garbage-collected.

In different embodiments of the invention, different events or knowledge may be cause for sticking the window bottom. For example, when an application, method or function is compiled or about to be compiled, the window bottom may be held in place until compilation is complete, particularly if the compilation is known to use or produce a large amount of temporarily data. As another example, when a graph or model is to computed or generated using a large amount of temporary data, the window bottom may be stuck in place until the data are no longer needed.

FIG. 1 is a block diagram of a generational garbage collection apparatus that employs a dynamic window and knowledge regarding anticipatable garbage,

according to one embodiment of the invention. Although just one young generation and one old generation are depicted in FIG. 1, the illustrated embodiment may be adapted for any number of generations of any sizes. The apparatus may be employed in virtually any device having a processor and memory (e.g., a telephone, a personal digital assistant, a computer).

Memory 102 comprises a section of memory allocated to a heap or other temporary storage pool. Virtually any type of data or memory objects may be stored in memory 102, for virtually any period of time. Garbage collector 104 is configured to perform garbage collection on memory 102 when and as needed. Memory allocator 106 is configured to allocate memory 102 as needed (e.g., for temporary storage of data). Thus, memory allocator 106 may maintain an allocation pointer and/or data structures as needed (not shown in FIG. 1). The functions of garbage collector 104 and memory allocator 106 may be combined.

Window 110 includes lower bound 112 and upper bound 114. When the apparatus of FIG. 1 is initialized, window 110 is positioned with lower bound 112 coinciding with the lower boundary of memory 102.

During one phase of operation of the apparatus (a "normal" allocation phase) window 110 slides upward to form a window over different parts of memory 102. Thus, window 110 defines a young generation of data objects stored in memory 102, while the portion of memory 102 below the window (i.e., between lower bound 112 and the lower bound of memory 102) defines an old generation.

In the first phase of operation, new data or data objects are stored at memory locations at or near the top of window 110. The young generation is garbage collected when the window is filled. Similarly, the old generation is garbage collected when it is filled.

However, a second phase of operation (a "temporary" phase) is initiated upon activation of a particular signal, alert, flag or other notification. For example, the garbage collection apparatus may be signaled by a Java Virtual Machine (JVM), a source code compiler, an interrupt service routine, executing application code or some other set of code. When this signal is received, lower bound 112 of window 110 is held in place at its current location or at another location. For example, the young generation may be garbage collected and then the window moved so that lower bound 112 is at the allocation point as of the time of the signal, and then the lower bound may be held. Or, the lower bound may be moved so that it is positioned near a newly stored set of data.

During this second phase of operation, upper bound 114 may be expanded to accommodate new memory contents, even while lower bound 112 is held in place. Thus, window 110 may grow in size during the second phase of operation.

In an embodiment of the invention, the second phase ends and the first phase of operation resumes when another signal is received or at some other time (e.g., at a certain time, when a timer expires). Illustratively, the young generation may be garbage collected in response to the signal (or some time after the signal), and then the size of the window may be adjusted to a default size or to the size it was before the most recent second phase of operation.

In the illustrated embodiment of the invention, the signal to begin the second phase of operation is issued when the issuer is confident or certain that data or objects that have recently been stored or that are about to be stored will be garbage in a finite and relatively short period of time (e.g., less than one minute). Thus, the first signal (to begin the second phase) may be issued by a JVM or compiler when it begins or is about to begin compiling a method, function or other series of code or compilable instructions. In general, some application or

system entity possesses contextual information regarding data being created, and can make a prediction as to how temporary the data are.

The second signal (to end the second phase) is issued when the stored data has become garbage. In the compilation example, the second signal would be issued when compilation is complete.

In other embodiments of the invention, other events or knowledge may cause the first and second signals to be issued. In general, when a relative large amount of data is to be stored for a finite period of time, the first signal is raised; the second signal follows when the data are no longer needed (i.e., they are garbage).

Thus, in an embodiment of the invention, a young generation window is dynamic in size and behavior. One end of the window (upper or lower, depending on how the window is configured and memory is filled) can be held in position for a period of time while the other end continues to move.

Implementation of an embodiment of the invention helps avoid over-promotion. Because the young generation is held in position for the useful lifetime of a (relatively large) set of data, those data are not promoted to an older generation, and the older generation is garbage collected less frequently. The young generation may also end up being garbage collected less frequently. In particular, when one end of the window is fixed in position and the second phase of operation of the window is implemented, the window may not be garbage collected until the first phase of operation ends.

FIG. 2 is a flowchart demonstrating a method of doing garbage collection with a dynamic window, according to one embodiment of the invention.

In operation 202, the first (or normal allocation) phase of garbage collection is active. That is, a window defining a young generation of memory contents is allowed to slide along a larger, older, generation of memory contents.

Either or both generations may be garbage collected as frequently as necessary (e.g., when the window or larger section of memory is filled). When the older generation, or full memory, is garbage collected and compacted, the window may fall to a position just above the compacted contents.

In operation 204, one or more data objects are to be stored in memory. The realization that data are being stored may be made by any set of code or firmware (e.g., part of an application program, a system routine, a BIOS, a JVM).

In operation 206, a determination is made as to whether to initiate the second (or temporary allocation) phase of operation. In this embodiment, this determination entails two examinations – of the amount of data to be stored, and the certainty that the data will become garbage within a finite or predetermined period of time.

The threshold amount of data that must be involved in order to trigger the second phase may be programmable, and may differ among different embodiments of the invention. A single data object may suffice in one embodiment, or an amount of data sufficient to fill a specified percentage of the memory (or young generation window) may be needed. The amount of time before the data will become garbage may be on the order of seconds, minutes or fractions thereof. If the thresholds are met, the method advances to operation 208. Otherwise, the method returns to operation 202.

In operation 208, the garbage collector is signaled or instructed to initiate a temporary allocation phase. The signal may be initiated by the processor or any other entity that possesses the knowledge that temporary data are being stored.

In operation 210, the garbage collector may perform a garbage collection on the young window or an older generation. Then, the young window may be positioned at the location (i.e., address) that the allocation point was at when the signal was received.

In operation 212, one end (e.g., the lower end) of the young window is fixed (i.e., not permitted to slide) and the garbage begins the temporary phase of memory allocation. The garbage collector may receive information regarding the amount of data to be stored and/or the amount of time (or an absolute time) after which the data are expected or certain to become garbage.

The window's other end may be made flexible, so that the window can continue to accept new data. The flexible end of the window may expand any number of times, to accommodate virtually any amount of data, during the temporary allocation phase.

In operation 214, garbage collection may be performed on the young and/or old generation as needed.

In operation 216, the garbage collector receives a second signal indicating that it may resume the first phase of operation, thus indicating that data in the young window are (or are expected to be) garbage. This signal may be self-generated (e.g., via a timer), or may be received from the same entity that issued the first signal.

In one alternative embodiment, the garbage collector may automatically halt second phase operations at a specified time or after a specified amount of time of operating in phase two. For example, second phase operations may be stopped after the young window grows to a threshold percentage (e.g., 50%, 100%) of the older generation or the entire memory. In another alternative embodiment, the second phase of operation may be aborted if the young window grows to or past a threshold size.

In operation 218, the young window is garbage collected and compacted, and the size of the window is readjusted to a default size. The position or location of the window may change, depending on how much of the young window

survives the garbage collection. Thus, the window may be adjusted to a position just above the compacted survivors of the garbage collection.

After operation 218, the illustrated method returns to operation 202 or ends.

An embodiment of the invention is well suited for implementation in a device with limited memory, such as a telephone. A telephone may have a few or several megabytes of memory, and may employ one-half to one megabyte as temporary storage (e.g., a heap) that is garbage collected.

The telephone may be configured with a JVM to interpret or execute various code (e.g., sounds, games, multimedia applications) that may generate temporary data. For example, a method or function of a program may be compiled by the JVM to speed its execution. The JVM may know that compilation will produce a significant amount of short-lived data. Thus, the JVM may instruct the garbage collection scheme to implement a temporary allocation phase of operation, and may signal for normal allocation to resume when compilation is complete. When temporary allocation initiated, the young window may be garbage collected and adjusted to have its lower bound at the address where the allocation point was when the signal was received to begin temporary allocation.

This is just one example of how a garbage collection scheme may obtain, or receive information from, outside the garbage collection environment, to increase the efficiency of garbage collection.

The foregoing embodiments of the invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the invention to the forms disclosed. Accordingly, the scope of the invention is defined by the appended claims, not the preceding disclosure.

The program environment in which a present embodiment of the invention is executed illustratively incorporates a general-purpose computer or a special purpose device such as a hand-held computer. Details of such devices (e.g., processor, memory, data storage, display) may be omitted for the sake of clarity.

It should also be understood that the techniques of the present invention may be implemented using a variety of technologies. For example, the methods described herein may be implemented in software executing on a computer system, or implemented in hardware utilizing either a combination of microprocessors or other specially designed application specific integrated circuits, programmable logic devices, or various combinations thereof. In particular, the methods described herein may be implemented by a series of computer-executable instructions residing on a suitable computer-readable medium. Suitable computer-readable media may include volatile (e.g., RAM) and/or non-volatile (e.g., ROM, disk) memory, carrier waves and transmission media (e.g., copper wire, coaxial cable, fiber optic media). Exemplary carrier waves may take the form of electrical, electromagnetic or optical signals conveying digital data streams along a local network, a publicly accessible network such as the Internet or some other communication link.